Redes Neuronales y Aprendizaje profundo

Tema 3 – Redes Neuronales Recurrentes

Irina Arévalo

Contenido

1. Procesamiento del Lenguaje Natural
2. Redes neuronales recurrentes
3. Mecanismos de atención

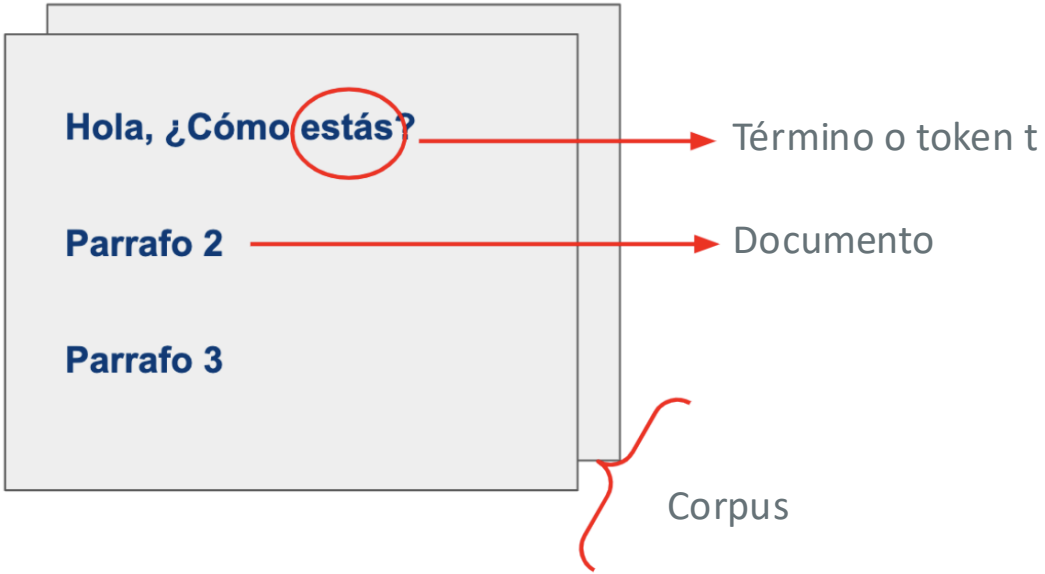
1. Procesamiento del Lenguaje Natural

- El Procesamiento del Lenguaje Natural (NLP) es el conjunto de técnicas basadas en métodos estadísticos o computacionales para comprender o procesar texto con el fin de resolver tareas determinadas
- En particular necesitamos diseñar algoritmos que permitan a los sistemas entender lenguaje natural (humano) para realizar la tarea.
- Este proceso no es trivial, ya que el lenguaje:
  - Es cultural
  - Es cambiante
  - Es multimodal
  - Depende de múltiples contextos

1. Procesamiento del Lenguaje Natural

- Vamos a trabajar con:
  - Documentos (una unidad de análisis de texto, por ejemplo un artículo de la Wikipedia, una frase…)
  - Corpus (un conjunto de documentos, por ejemplo toda la Wikipedia)
  - Token (secuencia de caracteres que se agrupa como unidad útil, por ejemplo una palabra)
  - Vocabulario (el conjunto de tokens posibles en un contexto particular, por ejemplo todas las palabras de la Wikipedia)
- También vamos a encontrarnos con dos tipos de conjuntos de datos:
  - No anotados, por ejemplo wikipedia, twitter, libros, etc.
  - Anotados, por ejemplo tweets con su sentimiento o respuestas de chatbots a preguntas
- Las anotaciones pueden ser muy costosas, porque requieren anotadores humanos, o poco costosas, porque se derivan automáticamente de los datos

1. Procesamiento del Lenguaje Natural



Para procesar esta información vamos a vectorizar las palabras/documentos después de un proceso de estandarización del código (Lab 4.1)

1. Procesamiento del Lenguaje Natural

One-hot encoding: por cada documento en el corpus se calcula un vector que representa si cada palabra del vocabulario aparece o no en ese documento

Hace buen día y
buen tiempo
No llueve
No llevo paraguas

Vocabulario: {“hace”, “buen”, “día”, “y”, “tiempo”, “no”, “llueve“, “llevo”, “paraguas}

Hace	Buen	Día	Y	Tiempo	No	Llueve	Llevo	Paraguas
1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	0	0
0	0	0	0	0	1	0	1	1

1. Procesamiento del Lenguaje Natural

Vectores de frecuencia: por cada documento en el corpus se calcula un vector que representa cuántas veces cada palabra del vocabulario aparece en ese documento

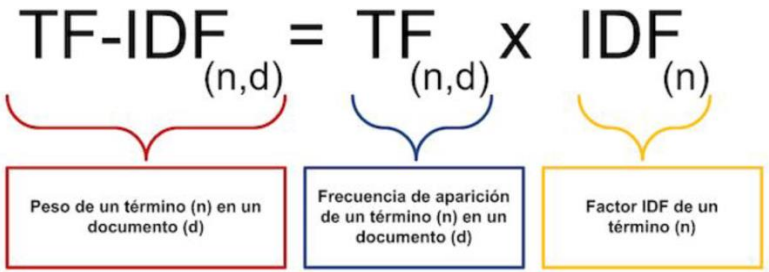
Hace buen día y
buen tiempo
No llueve
No llevo paraguas

Vocabulario: {“hace”, “buen”, “día”, “y”, “tiempo”, “no”, “llueve“, “llevo”, “paraguas}

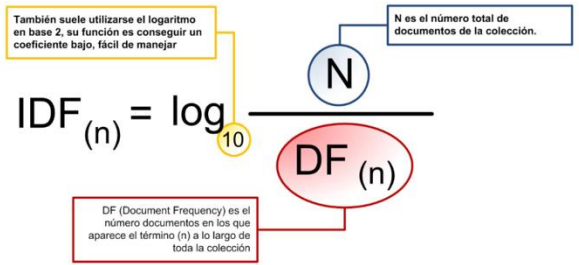
Hace	Buen	Día	Y	Tiempo	No	Llueve	Llevo	Paraguas
1	2	1	1	1	0	0	0	0
0	0	0	0	0	1	1	0	0
0	0	0	0	0	1	0	1	1

1. Procesamiento del Lenguaje Natural

TF-IDF (Term frequency-Inverse term frequency): cuantifica cómo de importante es un término en un documento



El término IDF es la proporción de documentos en el corpus que incluyen el término



1. Procesamiento del Lenguaje Natural

TF-IDF (Term frequency-Inverse term frequency)

Hace buen día y
buen tiempo
No llueve
No llevo paraguas

Vocabulario: {“hace”, “buen”, “día”, “y”, “tiempo”, “no”, “llueve“, “llevo”, “paraguas}

Hace	Buen	Día	Y	Tiempo	No	Llueve	Llevo	Paraguas
Log(4/1)	2*Log(4/1)	Log(4/1)	Log(4/1)	Log(4/1)	0	0	0	0
0	0	0	0	0	1*Log(4/2)	Log(4/1)	0	0
0	0	0	0	0	1	0	Log(4/1)	Log(4/1)

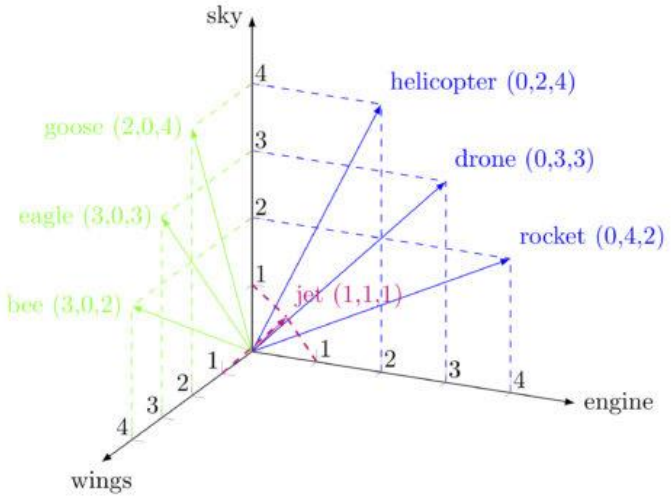
1. Procesamiento del Lenguaje Natural

- Cualquier modelo o vectorización de texto en don de el resultado no se modifique con el orden de las palabras es una representación bag-of-words o bolsa de palabras
- Estas representaciones no tienen en cuenta el contexto y dan la misma importancia a palabras muy distintas.

Document D1	*The child makes the dog happy* the: 2, dog: 1, makes: 1, child: 1, happy: 1
Document D2	*The dog makes the child happy* the: 2, child: 1, makes: 1, dog: 1, happy: 1

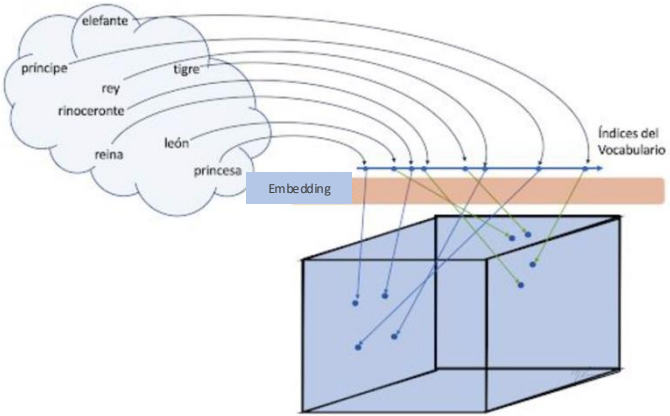
↓

	child	dog	happy	makes	the	BoW Vector representations
D1	1	1	1	1	2	[1,1,1,1,2]
D2	1	1	1	1	2	[1,1,1,1,2]



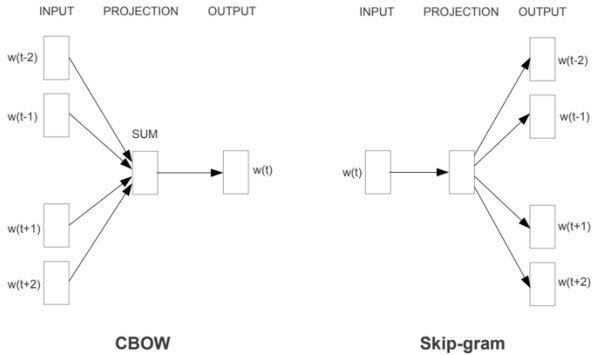
1. Procesamiento del Lenguaje Natural

- Para evitar estas representaciones poco densas se han creado los embeddings, representaciones numéricas densas de tamaño fijo de un dato
- Se usa sobre todo con texto, pero también se pueden mapear imágenes, entidades u observaciones a vectores

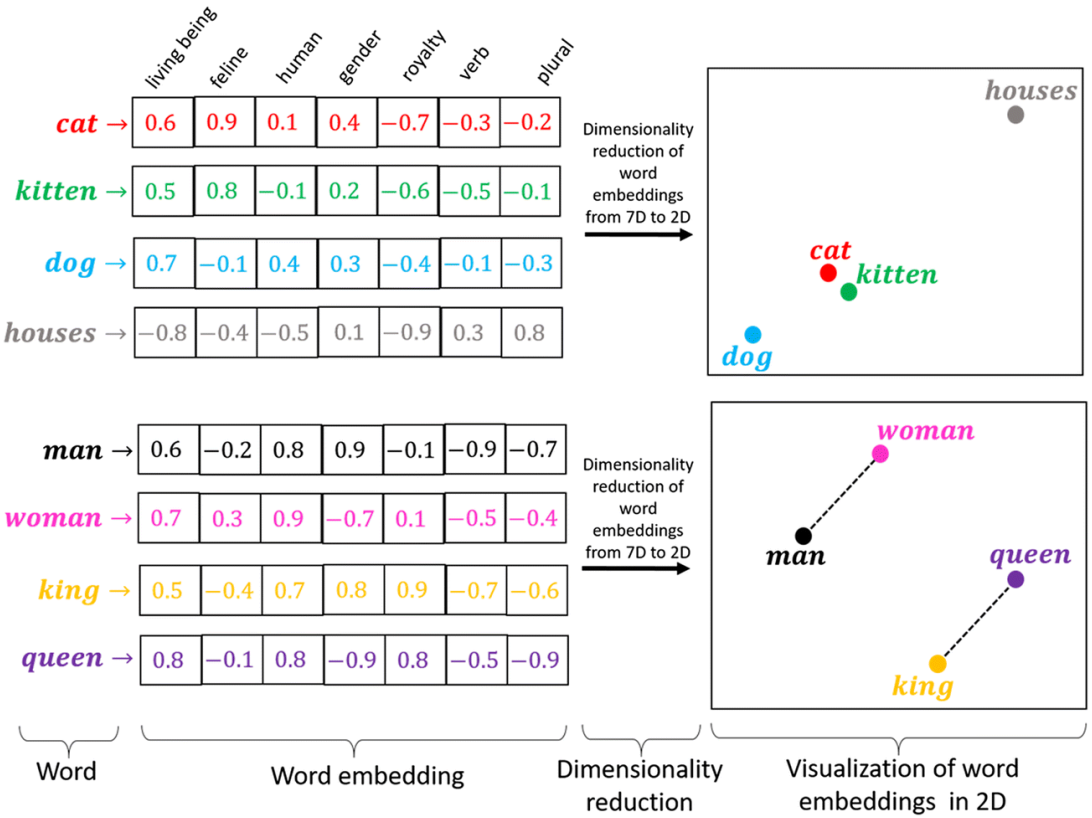


1. Procesamiento del Lenguaje Natural

- El modelo de embeddings de palabras más famoso y extenso es word2vec, una propuesta de Google en 2013 en el que proponen dos modelos para relacionar la palabra y el contexto
- En el modelo CBOW (Continuous Bag-of-Words) intentamos predecir una palabra en función de su contexto. Para ello se entrena un modelo usando pares (contexto, palabra target)
- En el modelo Skip-gram predecimos el contexto en función de la palabra. Por ejemplo si pasamos la palabra “navidad” nos devolverá “que”, “tengas” y “feliz”



1. Procesamiento del Lenguaje Natural



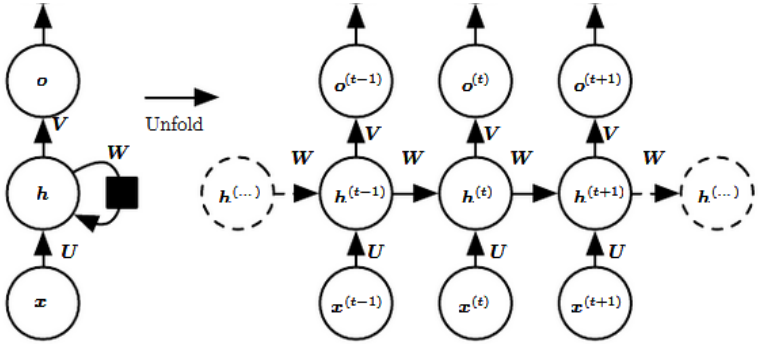
https://projector.tensorflow.org/

2. Redes neuronales recurrentes

- El principal inconveniente de las redes neuronales densas a la hora de trabajar con texto es que asumen una longitud fija de los vectores de entrada y salida, que se conoce de antemano

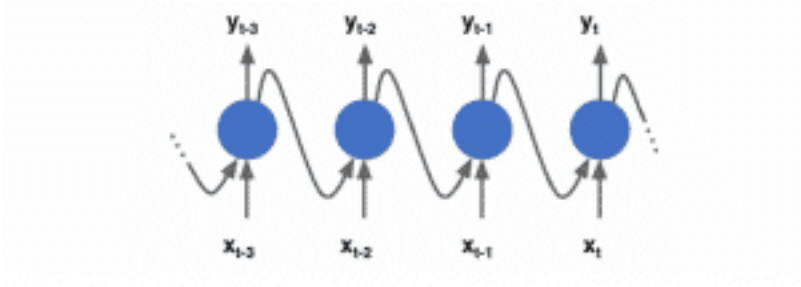
- Las redes neuronales recurrentes, o RNNs, son una familia especial de redes neuronales que fueron desarrolladas explícitamente para modelar datos secuenciales, como el texto.

- Las RNNs procesan una secuencia de palabras o letras $x^{(1)}, \ldots, x^{(t)}$ recorriendo sus elementos uno por uno y capturando información basada en los elementos anteriores. Esta información se almacena en estados ocultos $h^{(t)}$ que son la memoria de la red.



2. Redes neuronales recurrentes

- En cada instante de tiempo esta neurona recurrente recibe la entrada *x* de la capa anterior, así como su propia salida del instante de tiempo anterior para generar su salida *y*.



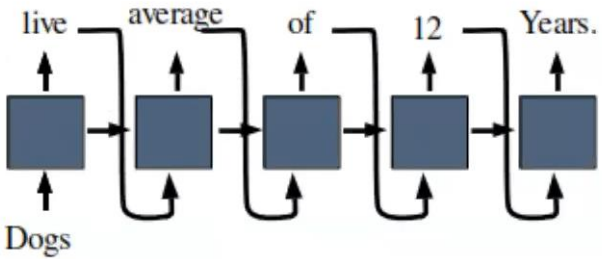
- En la capa de neuronas recurrentes en cada instante de tiempo cada neurona recibe dos entradas, la entrada correspondiente de la capa anterior y a su vez la salida del instante anterior de la misma capa.

2. Redes neuronales recurrentes

- Ahora cada neurona recurrente tiene dos conjuntos de parámetros, el que aplica a la entrada de datos que recibe de la capa anterior y el que aplica a la entrada de datos correspondiente al vector salida del instante anterior:

$$y_t = \sigma(W x_t + U y_{t-1} + b)$$

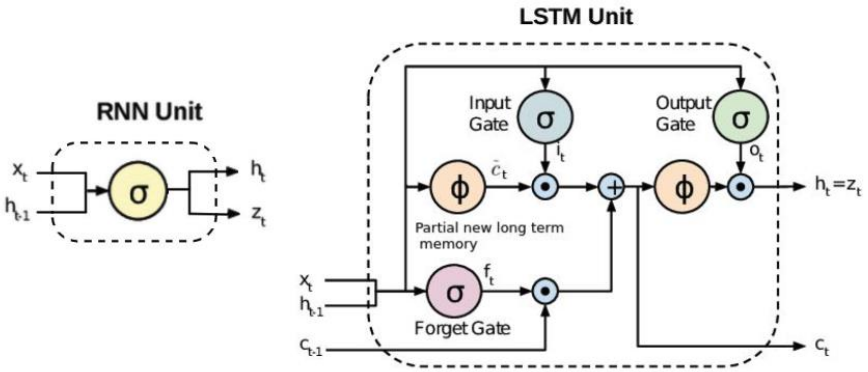
donde $x=(x_1,\ldots,x_T)$ representa la secuencia de entrada proveniente de la capa anterior, W los pesos de la matriz y b el bias como en las capas densas, y U es la matriz de pesos que opera sobre el estado de la red en el instante de tiempo anterior (y_{t-1}) anterior. Los pesos se actualizan en la fase de backpropagation



2. Redes neuronales recurrentes

- En el algoritmo de backpropagation los pesos se actualizan en base a las capas más cercanas. Por eso las RNN más sencillas no son capaces de aprender patrones muy extendidos en el tiempo, si no que solo son eficaces en rangos cortos, como por ejemplo secuencias de 10 elementos.

- Para mitigar el problema de *short-term memory* se creó un nuevo tipo de celda de memoria que sí es capaz de extraer patrones de secuencias de mayor longitud, llamada capa de Memorias de Corto y Largo Plazo (LSTM, por sus siglas en inglés de *Long Short Term Memory*)



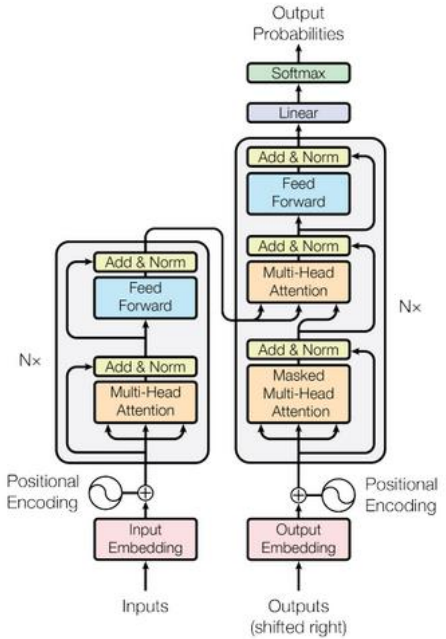
2. Redes neuronales recurrentes

- En una neurona LSTM hay tres puertas: de entrada (input gate), de olvidar (forget gate) y de salida (output gate). Estas puertas determinan si se permite o no una nueva entrada, se elimina la información porque no es importante o se deja que afecte a la salida en el paso de tiempo actual

- Hay otras arquitecturas avanzadas de RNN como es la Gated Recurrent Unit (GRU), que usan el mismo principio que LSTM, pero están simplificadas de manera que su rendimiento está a la par con LSTM pero computacionalmente son más eficientes.

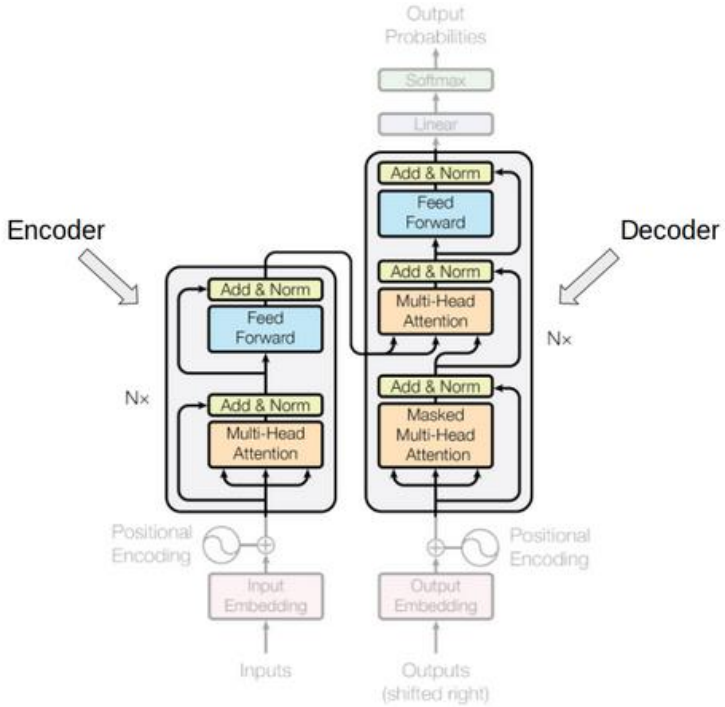
3. Mecanismos de atención

- Los modelos más modernos de NLP como BERT o GPT usan Transformers, una arquitectura basada en mecanismos de atención. Estas capas de atención codifican cada palabra de una frase en función del resto de la secuencia, permitiendo así introducir el contexto en la representación matemática del texto, motivo por el cual a los modelos basados en Transformer se les denomina también Embeddings Contextuales.



3. Mecanismos de atención

- El Transformer se basa en un Encoder y un Decoder. La idea clave aquí es que se utiliza el encoder para analizar el contexto de la secuencia de entrada y el decoder es el encargado de generar la secuencia del output a partir de este contexto.



3. Mecanismos de atención

- El encoder suele tener tres partes:
  - Capa de embedding del input: convierte el input en los embeddings
  - RNN encoder: los embeddings se pasan por la RNN (habitualmente LSTM o GRU) para aprender el contexto
  - Vector de contexto: el estado oculto final de la RNN, que resume la sucesión del input, es el resultado del encoder.

- El decoder toma el vector de contexto y genera un resultado. Suele tener las siguientes partes:
  - Capa de embedding del resultado: al igual que la capa de embedding del input, la sucesión del output se representa como un vector denso
  - RNN decoder: los embeddings del output se pasan por la RNN del decoder, que genera el output teniendo en cuenta lo generado anteriormente y el vector de contexto como el estado oculto inicial
  - Capa de resultado: dependiendo del problema (traducción, siguiente palabra…) esta capa da forma al resultado

3. Mecanismos de atención

- Esta es la arquitectura que usan los grandes modelos de lenguaje (LLMs) para generar texto, como GPT-4, Gemini, BERT...

- En ese caso, el modelo es entrenado de manera no supervisada en grandes cantidades de datos textuales. Durante el entrenamiento, el modelo aprende a predecir la siguiente palabra en una secuencia dada una parte del texto anterior. Este proceso de pre-entrenamiento ayuda al modelo a capturar patrones lingüísticos complejos y construir representaciones contextualizadas de las palabras.

Redes Neuronales y Aprendizaje profundo

Tema 3 – Redes Neuronales Recurrentes

Irina Arévalo